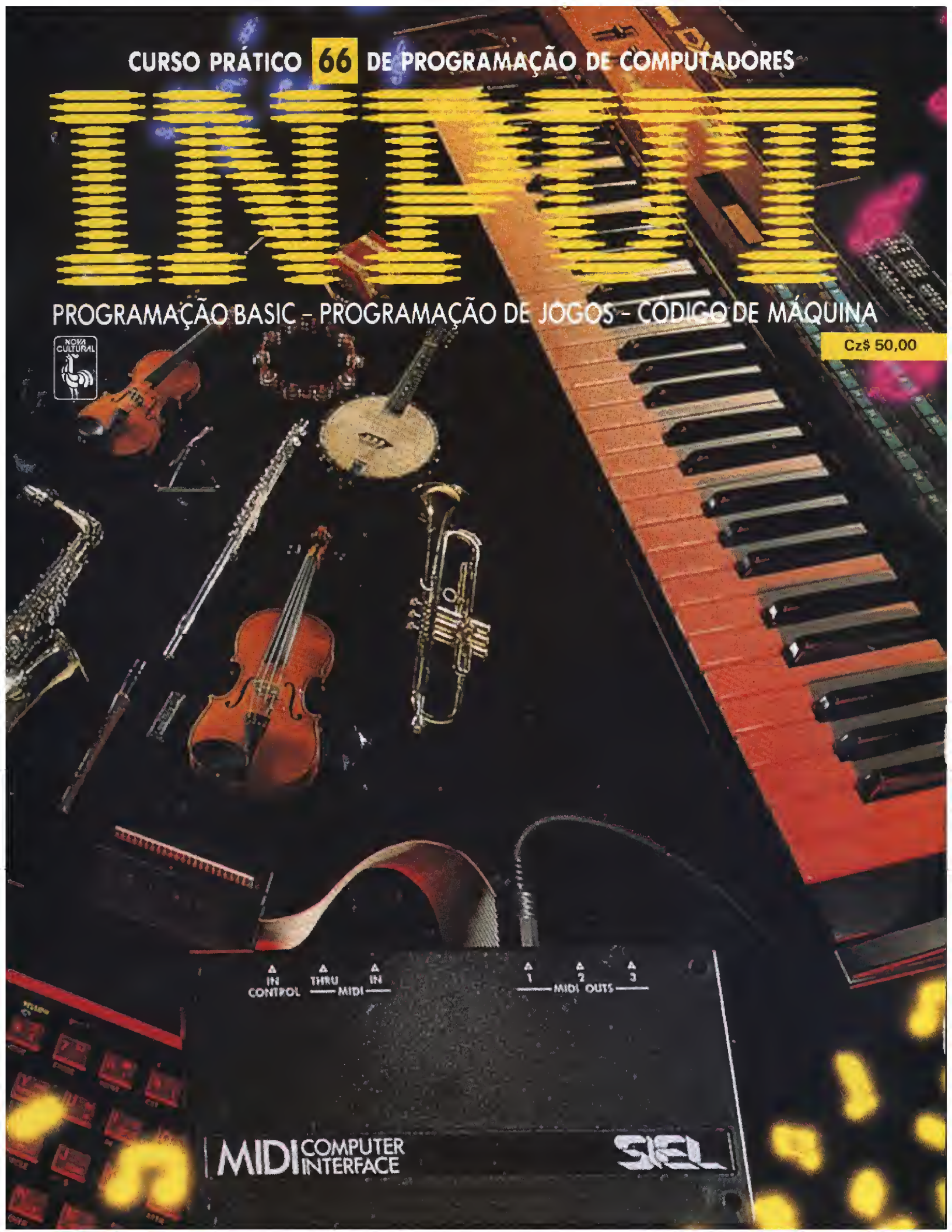
CURSO PRÁTICO 66 DE PROGRAMAÇÃO DE COMPUTADORES

MIDI

PROGRAMAÇÃO BASIC – PROGRAMAÇÃO DE JOGOS – CÓDIGO DE MÁQUINA

NOVA CULTURAL

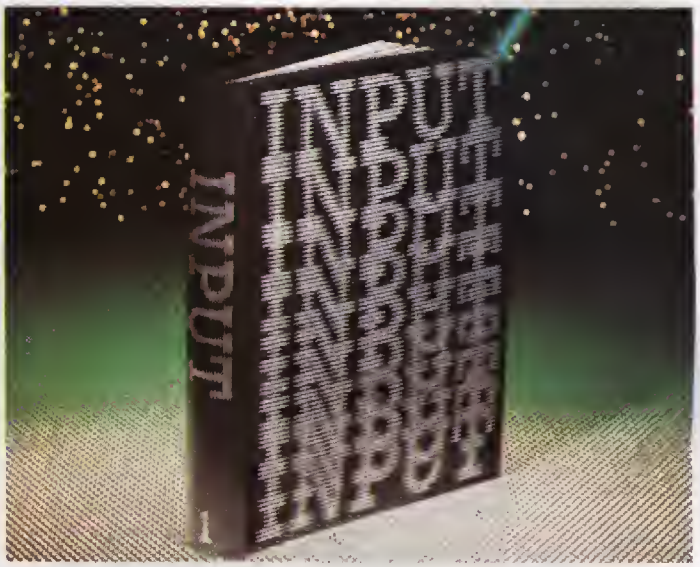
Cz$ 50,00



INPUT

Vol. 5 N.º 66

NESTE NÚMERO



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: 1. PESSOALMENTE — Por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em **São Paulo**, os endereços são: rua Brigadeiro Tobias, 773, Centro; avenida Industrial, 117, Santo André; e no **Rio de Janeiro**: avenida Mem de Sá, 191/193, Centro. 2. POR CARTA — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidora Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132, Jardim Teresa — CEP 06000 — Osasco — SP. Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na agência do Correio. 3. POR TELEX — Utilize o nº (011) 33 670 DNAP.

Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações, Lda. — Qta. Pau Varais, Azinhaga de Fetais — 2 685, Camarate — Lisboa; Apartado 57 — Telex 43 069 JARLIS P.

Atenção: Após seis meses do encerramento da coleção, os pedidos serão atendidos dependendo da disponibilidade do estoque.

Obs.: Quando pedir livros, mencione sempre título e/ou autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

EDITOR
RICHARD CIVITA

NOVA CULTURAL

Presidente
Flávio Barros Pinto
Diretoria
Carmo Chagas, Iara Rodrigues, Pierluigi Bracco, Plácido Nicoleto, Roberto Silveira, Shozi Ikeda, Sônia Carvalho

REDAÇÃO
Diretor Editorial: Carmo Chagas
Editores Executivos:
Antonio José Filho, Berta Sztark Amar
Editor Chefe: Paulo de Almeida
Editoras Assistentes: Ana Lúcia B. de Lucena, Marisa Soares de Andrade
Chefe de Arte: Carlos Luiz Batista

Assistentes de Arte: Dagmar Bastos Sampaio, Grace Alonso Arruda, Monica Lenardon Corradi

Colaboradores
Consultor Editorial Responsável:
Dr. Renato M. E. Sabbatini
(Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda., Campinas, SP
Tradução, adaptação, programação e redação:
Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Marcos Huascar Velasco, Raul Neder Porrelli, Ricardo J. P. de Aquino Pereira
Coordenação Geral: Rejane Felizatti Sabbatini

COMERCIAL
Diretor Comercial: Roberto Silveira
Gerente Comercial: Flávio Maculan
Gerente de Circulação: Denise Mozol
Gerente de Propaganda e Publicidade: José Carlos Madio
Gerente de Pesquisa e Análise de Mercado:
Wagner M. P. Nabuco de Araújo

CLC

A Editora Nova Cultural Ltda. é uma empresa do Grupo CLC — Comunicações, Lazer e Cultura
Presidente: Richard Civita
Diretoria: Flávio Barros Pinto, João Gomez, Menahen M. Politi, Renê C. X. Santos, Stélio Alves Campos

Edição organizada pela Editora Nova Cultural Ltda.
Av. Brigadeiro Faria Lima, nº 2000 - 3º andar
CEP 01452 - São Paulo - SP - Brasil
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

DIVERTIMENTOS MATEMÁTICOS

- TIPOS DE QUEBRA-CABEÇA
- USANDO O COMPUTADOR
- SISTEMAS DE EQUAÇÕES
- TENTATIVA E ERRO
- TRUQUES MATEMÁTICOS

Desafie seu microcomputador a decifrar alguns quebra-cabeças matemáticos. Você também irá aprender técnicas de grande utilidade na resolução de sistemas de equações.

Ao chegar a este ponto de *INPUT*, você já teve contato com as principais técnicas de programação em BASIC. Dado um problema, será bem capaz de construir um programa para solucioná-lo (caso haja uma solução, é claro). Entretanto, a não ser que você tenha um *hobby* ou esteja trabalhando em algum projeto, são raras as suas oportunidades de mostrar o que sabe fazer em seu microcomputador.

Certos quebra-cabeças representam gostosos desafios para quem quer praticar programação ou simplesmente exercitar o raciocínio. A popularidade desse tipo de divertimento não é novidade. Os egípcios destacavam-se por sua habilidade em decifrar enigmas, assim como os gregos eram admirados pelos seus interessantes quebra-cabeças lógicos e matemáticos e seus paradoxos inexplicáveis. De fato, muitos problemas inicialmente tratados como simples recreação transformam-se em grandes descobertas científicas.

Encontrar uma solução elegante para um determinado problema geralmente traz uma satisfação muito grande. Mas, para os usuários de microcomputadores, a resolução de quebra-cabeças constitui, antes de tudo, um excelente exercício de programação — na verdade, bem mais eficiente que a análise e execução de programas prontos, uma vez que os estimula a escolher técnicas e aplicar idéias próprias.

QUEBRA-CABEÇAS NO COMPUTADOR

Existem diversos tipos de quebra-cabeça, e nem todos podem ser resolvidos no computador. Alguns exigem simplesmente intuição ou um pouco de raciocínio lógico. Um exemplo clássico seria o seguinte: um homem possui um lobo,


47
93
71
1
2

um carneiro e um repolho, e quer levá-los para o outro lado do rio em uma canoa. A canoa é muito pequena e ele só pode transportar um de cada vez. Mas não deve deixar sozinhos em qualquer das margens do rio nem o lobo e o carneiro, nem o carneiro e o repolho, pois algum não sobraria... Como chegar ao outro lado do rio com todos os seus bens?

Você pode resolver esse problema utilizando um computador (se quiser, escreva um programa), mas é muito mais rápido encontrar a sua solução com um lápis e um papel.

Os problemas mais adequados ao uso do computador são aqueles em que, dado um certo número de situações com os respectivos valores, pergunta-se sobre o valor de uma situação hipotética. A máquina também é útil quando a questão envolve cálculos aritméticos complexos ou geometria. Nesses casos, chega-se mais depressa ao resultado escrevendo um programa do que buscando a solução à mão.

Neste artigo, mostraremos como resolver três dos tipos mais comuns de quebra-cabeça. Antes de olhar as soluções, tente encontrá-las sozinho. Em seguida, examine os programas e veja como eles funcionam.

SIMPLIFICANDO O PROBLEMA

Se você não está acostumado a resolver quebra-cabeças, poderá ter algumas dificuldades em compreender o que é pedido em cada um deles, pois a maioria requer certa prática em extrair as informações essenciais de um texto razoavelmente confuso.

O primeiro tipo de problema, por exemplo, apresenta um texto de tamanho considerável que o torna muito mais complicado do que na realidade é. Vamos começar por uma situação bastante simples, que pode ser resolvida sem o auxílio do computador.

Em uma certa manhã de inverno, um grupo de amigos entra em uma lanchonete e toma três xícaras de café e duas de chá, pagando uma conta no valor de Cz$ 44,00. No dia seguinte, eles retornam ao mesmo lugar, mas tomam o dobro de xícaras de chá e um café a menos. Sabendo que desta vez a conta foi de Cz$ 48,00, qual é o preço de cada xícara de chá?

Se você conseguir remover todas as informações necessárias e colocar os dados principais sob a forma de variáveis, o problema consistirá na solução das seguintes equações: $3c + 2h = 44$ e $2c + 4h = 48$.

Estamos diante de um sistema de equações lineares: ambas devem ser resolvidas com os mesmo valores de **c** e **h**, que não estão elevados a nenhuma potência ($3c\uparrow 2 - 2h = 44$, por exemplo, não seria uma equação linear). Para se chegar a uma solução desse sistema são necessárias tantas equações quantas forem as variáveis. Neste caso, existem dois valores desconhecidos — **c** e **h** — e duas equações; logo, existe uma solução.

Uma maneira de resolver o problema seria isolar uma variável na segunda equação — $c = (48 - 4h)/2$ — e substituí-la na primeira, obtendo: $3 (48 - 4h)/2\ 2h = 44$. Como resultado, teremos $h = 7$, ou seja, uma xícara de chá custa Cz$ 7,00.

Sistemas com duas equações são bem fáceis de resolver. Com três, também não são tão difíceis. Mas, a partir daí, você, certamente, irá precisar da ajuda do computador.

O VENDEDOR DE SELOS

Um negociante possui uma caixa de selos estrangeiros agrupados em seis tipos de envelope. Cada envelope, classificado de **A** a **F**, tem um preço diferente. Seis crianças, membros de um clube filatélico, gastaram todo o seu dinheiro do seguinte modo:

	A	B	C	D	E	F	Preço
Belinha	6	2	3	1	1	2	Cz$ 341
Nanda	14	11	0	1	2	1	Cz$ 469
Digo	0	1	3	6	4	3	Cz$ 598
Lu	5	3	5	2	1	1	Cz$ 376
Dudoca	12	1	4	4	3	2	Cz$ 587
Alcino	8	0	1	1	0	3	Cz$ 293

Qual é o preço de cada envelope?

Poderíamos resolver esse sistema eliminando uma variável por vez, mas, agindo assim, perderíamos muito tempo e qualquer erro aritmético inutilizaria o resultado.

Existem, porém, diversas maneiras de se resolver um sistema de equações — você mesmo seria capaz de inventar um método diferente. O que escolhemos para nosso programa é bem mais rápido e relativamente simples.

```
10 INPUT "DIGITE NUMERO DE LI
NHAS ";R
15 LET C=R+1
20 DIM A(R,C): DIM B(R,C):
DIM A$(C-1,20)
30 FOR K=1 TO C-1
40 INPUT "NOMES DAS COLUNAS "
,A$(K)
 50 NEXT K
 60 FOR J=1 TO R
 70 PRINT : PRINT "VALORES PAR
 A A LINHA ";J
 80 FOR K=1 TO C
 90 INPUT A(J,K)
 95 PRINT A(J,K);" ";
100 LET B(J,K)=A(J,K)
110 NEXT K: NEXT J
120 FOR L=1 TO R
130 GOSUB 230
140 GOSUB 280
150 NEXT L
160 CLS
170 FOR K=1 TO C-1: PRINT AT 1
,4*K-4;A$(K): NEXT K
180 FOR J=1 TO R: FOR K=1 TO C
: PRINT AT 1+J,K*4-4;B(J,K)
190 NEXT K: NEXT J
200 PRINT "RESPOSTAS:"
210 FOR K=1 TO C-1: PRINT AT 4
+R,K*4-4;A(K,C): NEXT K
220 STOP
240 LET D=A(L,L)
250 FOR K=1 TO C
260 LET A(L,K)=A(L,K)/D
270 NEXT K: RETURN
290 FOR J=1 TO R
300 IF J=K THEN  NEXT J:
RETURN
310 LET F=A(J,L)
320 FOR K=1 TO C
330 LET A(J,K)=A(J,K)-F*A(L,K)
340 NEXT K: NEXT J: RETURN
```

```
10 CLS:INPUT"DIGITE NUMERO DE L
INHAS E COLUNAS ";R,C
20 DIM A(R,C),B(R,C),A$(C-1)
30 FOR K=1 TO C-1
40 INPUT"NOMES DAS COLUNAS ";A$
(K)
50 NEXT
60 FOR J=1 TO R
70 PRINT"VALORES PARA AS LINHAS
 ";J
80 FOR K=1 TO C
90 INPUT A(J,K)
100 B(J,K)=A(J,K)
110 NEXT K,J
120 FOR L=1 TO R
130 GOSUB 230
140 GOSUB 280
150 NEXT
160 CLS
170 FOR K=1 TO C-1:PRINT @4*K-3
,A$(K):NEXT
180 FOR J=1 TO R:FOR K=1 TO C:P
RINT @4*K-5+32*J,B(J,K)
190 NEXT K,J
200 PRINT"RESPOSTAS:"
210 FOR K=1 TO C-1:PRINT A$(K);
" = ";:PRINT USING"######.##";A
(K,C):NEXT
220 END
230 D=A(L,L)
250 FOR K=1 TO C
260 A(L,K)=A(L,K)/D
270 NEXT:RETURN
280 FOR J=1 TO R
300 IF J=L THEN NEXT:RETURN
```

```
310 F=A(J,L)
320 FOR K=1 TO C
330 A(J,K)=A(J,K)-F*A(L,K)
340 NEXT:NEXT:RETURN
```

```
10  HOME : INPUT "DIGITE O NUME
RO DE LINHAS E COLUNAS ";R,C
20  DIM A(R,C): DIM B(R,C): DIM
 A$(C - 1)
30  FOR K = 1 TO C - 1
40  INPUT "ENTRE COM OS NOMES P
ARA ASCOLUNAS ";A$(K)
50  NEXT K
60  FOR J = 1 TO R
70  PRINT : PRINT "INTRODUZA VA
LORES PARA A LINHA ";J
80  FOR K = 1 TO C
90  INPUT A(J,K)
100 B(J,K) = A(J,K)
110  NEXT K: NEXT J
120  FOR L = 1 TO R
130  GOSUB 240
140  GOSUB 290
150  NEXT L
160  HOME
170  FOR K = 1 TO C - 1: VTAB (
1): HTAB (K * 5): PRINT A$(K):
NEXT K
180  FOR J = 1 TO R: FOR K = 1
TO C: HTAB (K * 5): VTAB (4 * J
): PRINT B(J,K)
190  NEXT K: NEXT J
200  VTAB (4 * J): HTAB (1): PR
INT "RESPOSTAS:-"
210  FOR K = 1 TO C - 1: VTAB (
4 * J + 4): HTAB (K * 5): PRINT
  INT (A(K,C) * 100) / 100: NEX
T K
220  END
240 D = A(L,L)
250  FOR K = 1 TO C
260 A(L,K) = A(L,K) / D
270  NEXT K: RETURN
290  FOR J = 1 TO R
300  IF J = L THEN  NEXT J: RET
URN
310  LET F = A(J,L)
320  FOR K = 1 TO C
330 A(J,K) = A(J,K) - F * A(L,K
)
340  NEXT K: NEXT J: RETURN
```

```
10 CLS:INPUT "DIGITE O NUMERO D
E LINHAS E COLUNAS ";R,C
20 DIM A(R,C):DIM B(R,C):DIM AS
(C-1)
30 FOR K=1 TO C-1
40 INPUT "ENTRE COM OS NOMES PA
RA AS COLUNAS ";A$(K)
50 NEXT K
60 FOR J=1 TO R
70 PRINT:PRINT "INTRODUZA VALOR
ES PARA A LINHA ";J
80 FOR K=1 TO C
90 INPUT A(J,K)
100 B(J,K)=A(J,K)
110 NEXT K,J
120 FOR L=1 TO R
130 GOSUB 240
140 GOSUB 290
150 NEXT L
160 CLS
170 FOR K=1 TO C-1:LOCATE K*5+1
,1:PRINT A$(K):NEXT K
180 FOR J=1 TO R:FOR K=1 TO C:L
OCATE K*5,4*J:PRINT B(J,K)
190 NEXT K,J
200 LOCATE 1,4*J:PRINT "RESPOST
AS:-"
210 FOR K=1 TO C-1:LOCATE K*5,4
*J+4:PRINT INT(A(K,C)*100)/100:
NEXT K
220 END
240 D=A(L,L)
250 FOR K=1 TO C
260 A(L,K)=A(L,K)/D
270 NEXT K:RETURN
290 FOR J=1 TO R
300 IF J=L THEN NEXT J:RETURN
310 F=A(J,L)
320 FOR K=1 TO C
330 A(J,K)=A(J,K)-F*A(L,K)
340 NEXT K,J:RETURN
```

A primeira parte do programa, da linha 10 à 110, permite que você introduza as informações, enquanto a segunda parte, da linha 120 à 150, chama as subrotinas de cálculo. A resposta será fornecida pela última parte, da linha 160 à 220.

Os valores são colocados em uma matriz **A(J,K)**, uma linha por vez. Neste programa, **J** se refere à linha da matriz e **K** corresponde à coluna. A matriz **A(J,K)** é copiada em outra idêntica — **B(JK)**. Conservamos, desse modo, a matriz original, que será impressa com a resposta, na linha 180.

O cálculo, feito linha por linha, é controlado pelo laço entre as linhas 170 e 210. Primeiro, a rotina entre as linhas 240 e 270 (230 e 270 no TRS-Color) seleciona o elemento de cada linha que pertence à diagonal da matriz (**A(1,1)**, **A(2,2)**, e assim por diante). Em seguida, divide cada um desses elementos pelo seu valor. Conseqüentemente, todos os elementos da diagonal da matriz tornam-se iguais a 1.

A próxima rotina executa a maior parte do trabalho. Embora tenha apenas seis linhas, é muito difícil entender o seu funcionamento.

Suponl amos que a rotina anterior já tenha feito **A(1,1) = 1**. O programa será então desviado para a linha 290 (280 no TRS-Color), com L = 1. A linha 300 não deixará que a linha 1 da matriz seja processada; assim, o laço partirá da linha 2. A linha 310 tomará o primeiro elemento da linha 2, colocando seu valor em **F**. Ainda em 2, a linha 320 irá selecionar cada coluna para que a linha 330 multiplique **F** pelo elemento da linha 1 da coluna em questão e subtraia esse resultado do elemento da linha 2 da coluna. O mesmo será feito com as linhas 3, 4, 5 e 6. Em seguida, o processo se repetirá com **L** = 2.

Ao final, os elementos pertencentes à diagonal da matriz continuarão iguais a 1, e os restantes, exceto os que se encontram na última coluna da direita, serão iguais a 0.

Será possível, então, ler diretamente os valores de **A**, **B**, **C** etc. na coluna não nula. Todos os microcomputadores, menos o Spectrum, apresentarão os resultados com duas casas decimais após a vírgula. Sem isso, uma resposta que seria 10 poderia aparecer como 9.999998 ou 10.000001.

O PRESENTE DE NATAL

Utilizando o programa anterior, você poderá resolver qualquer sistema de equações lineares que possua um número igual de equações e de variáveis. Muitos problemas, porém, incluem menos equações e apresentam incógnitas elevadas a algum expoente. Nesse caso, o melhor caminho para se chegar a uma resposta consiste no emprego da tentativa e erro. Geralmente, o próprio texto fornece alguma pista para encurtar seu trabalho.

Tente resolver este problema: no Natal, vovô Alberto, um excêntrico matemático, anunciou a seus dois jovens netinhos que cada um deles ganharia tantos pacotes quantos anos tivessem (contando apenas anos completos). Disse também que cada pacote conteria um número de envelopes correspondente à sua idade e que, em cada envelope, eles encontrariam, do mesmo modo, o valor em cruzados equivalente à idade. Por fim, comentou que, no ano seguinte, o mesmo presente lhe custaria Cz$ 500,00 a mais. Quantos anos têm seus netos?

Definindo a idade dos garotos como **A** e **B**, e o valor gasto no ano em questão como **M**, reduzimos o problema a duas equações:

$$A\uparrow 3 + B\uparrow 3 = M \text{ e}$$
$$(A + 1)\uparrow 3 + (B + 1)\uparrow 3 = M + 500$$

Temos, portanto, duas equações e três incógnitas. Sem recorrer à ajuda do computador, levaríamos um bom tempo experimentando valores de **A** e **B** que satisfizessem as duas equações. O computador trabalhará da mesma maneira — só que mais rapidamente e sem risco de cometer erros.

Antes de partir para os cálculos, devemos verificar se há no texto alguma informação que nos permita delimitar os

valores de **A** e **B**. A referência aos "jovens netinhos" sugere que os garotos não têm mais de catorze anos nem menos de três.

Como você poderá constatar, o programa para resolver o quebra-cabeça é bem simples.

```
10 FOR A=3 TO 14
20 FOR B=A TO 14
30 LET M=A^3+B^3
40 LET N=(A+1)^3+(B+1)^3
50 IF ABS (M+500-N)<.01 THEN
PRINT "A=";A,"B=";B
60 NEXT B
70 NEXT A
```

```
10 FOR A=3 TO 14
20 FOR B=A TO 14
30 LET M=A^3+B^3
40 LET N=(A+1)^3+(B+1)^3
50 IF ABS(M+500-N)<.01 THEN PRI
NT "A= ";A,"B= ";B
60 NEXT B
70 NEXT A
```

Embora o sistema possa apresentar mais de uma solução, obteremos apenas uma, devido à restrição introduzida nas linhas 10 e 20. Se o problema não especificasse, em sua formulação, "jovens netinhos", o laço **FOR...NEXT** seria bem mais extenso.

A função **ABS** da linha 50 evita que se cometa erros de arredondamento, checando se realmente **M + 500** é igual a **N**.

O método utilizado funciona para todos os problemas em que o número de incógnitas é maior que o número de equações. Outras situações podem exigir mais variáveis ou mais condições **IF...THEN**. Dependendo da complexidade do problema, o computador poderá levar alguns minutos ou então até horas para fornecer uma resposta, mas com certeza chegará a ela.

"MATEMÁGICA"

O terceiro grupo de quebra-cabeças que aparece em revistas especializadas em computador é o que envolve exclusivamente números. À primeira vista, parece simples questões aritméticas; mas, quando se tenta solucioná-las, logo se percebe a necessidade de recorrer à máquina.

Seguem-se alguns exemplos.

Existe um número formado de quatro dígitos que, quando invertido e multiplicado por outro número inteiro, retorna ao seu valor original. O problema consiste em descobrir se outros números apresentam essa mesma característica e, em caso afirmativo, apontar quais são eles.

Outro caso muito interessante é o do número 987654321. Ele é divisível por 17 e inclui todos os dígitos, de 1 a 9. Aqui, é preciso identificar um outro número que apresente essas mesmas propriedades.

Para solucionar ambos os problemas, devemos tratar o número como uma coleção de dígitos (e não como um valor numérico). Podemos dividir o número em unidades, dezenas, centenas etc. ou manipulá-lo como uma cadeia de caracteres, usando **RIGHT$**, **MID$** e **LEFT$** ou os comandos equivalentes no micro da linha Spectrum.

Resolveremos o problema do número de quatro dígitos empregando o primeiro método. Sendo tal número **ABCD**, chegaríamos a esta equação:

1000*A + 100*B + 10*C + D = X* (1000*D + 100*C + 10*B + A)

Como sempre, a maior dificuldade para encontrar as cinco incógnitas reside na delimitação do intervalo em que pode estar cada uma delas. **A** varia entre 1 e 9 — se fosse 0, teríamos um número de três dígitos. **B** e **C**, por sua vez, variam entre 0 e 9. O fator de multiplicação **X** não deve ser menor que 2 ou maior que 10/**D**, pois, nesse caso, a segunda metade da equação seria um número de cinco dígitos. Por essa mesma razão, **D** não pode ser maior que 4. Tendo delimitado esses intervalos, podemos escrever o programa para solucionar o problema.

```
10 FOR A=1 TO 9
20 FOR B=0 TO 9
30 FOR C=0 TO 9
40 FOR D=1 TO 4
50 FOR X=2 TO INT (9.9/D)
60 LET J=1000*A+100*B+10*C+D
70 LET K=1000*D+100*C+10*B+A
80 IF X*K=J THEN  PRINT J;"="
;X;"*";K
90 NEXT X: NEXT D: NEXT C:
NEXT B: NEXT A
```

```
10 FOR A=1 TO 9
20 FOR B=0 TO 9
30 FOR C=0 TO 9
40 FOR D=1 TO 4
50 FOR X=2 TO INT(9.9/D)
60 J=1000*A+100*B+10*C+D
70 K=1000*D+100*C+10*B+A
80 IF X*K=J THEN PRINT J;"=";X;
"*";K
90 NEXT X,D,C,B,A
```

Digite o programa e execute-o. Seja paciente: você precisará esperar um bom tempo para obter o resultado, pois o computador irá verificar todas as combinações possíveis.

Para solucionar o segundo problema, trataremos o número como se fosse uma cadeia de caracteres. O programa para o Spectrum é um pouco diferente dos programas destinados aos demais computadores, porque esse micro só manipula oito dígitos por vez, mas os princípios básicos são os mesmos.

```
 10 LET M=98765432
 20 LET M=M-27
 30 LET M$=STR$ M
 40 LET F=0
 50 FOR P=2 TO 9
 60 LET P$=STR$ P
 70 LET X=0: FOR K=1 TO 8: IF
 P$=M$(K TO K) THEN  LET X=K
 75 NEXT K
 80 IF X=0 THEN  LET F=F+1
 90 NEXT P: IF F=0 THEN  PRINT
 M$: STOP
100 GOTO 20
```

```
10 M=987654321
20 M=M-17
30 M$=STR$(M)
40 F=0
50 FOR P=1 TO 9
60 P$=CHR$(48+P)
70 X=INSTR(M$,P$)
80 IF X=0 THEN F=F+1
90 NEXT
100 IF F=0 THEN PRINT M$:END
110 GOTO 20
```

```
10 M = 987654321
20 M = M - 17
30 M$ =  STR$ (M)
40 F = 0
50  FOR P = 1 TO 9
60 P$ =  CHR$ (48 + P)
65  FOR Z = 1 TO  LEN (M$)
70  IF  MID$ (M$,Z,1) = P$ THEN
 90
80  NEXT Z:F = F + 1
90  NEXT P: IF F = 0 THEN  PRIN
T M$: END
100  GOTO 20
```

Fazendo uma contagem regressiva a partir de 987654321, o programa converte todo múltiplo de 17 em uma cadeia **M$**. As linhas 50 e 60 tratam cada um dos dígitos, de 1 a 9, como um caractere. A linha 70 verifica se o dígito está na cadeia **M$**; se não estiver, um indicador **F** é incrementado. A cadeia **M$** só será exibida se possuir todos os dígitos de 1 a 9.

MÚSICA, MICROS E MIDI

A interface MIDI oferece excelentes recursos para quem quer fazer música — amadorística ou profissionalmente. Veja como transformar seu computador em um sintetizador musical.

Embora seja um atributo relativamente recente dos micros, a capacidade de executar efeitos sonoros e musicais tem se tornado cada vez mais comum nos computadores pessoais de última geração. Muitos usuários com inclinações musicais chegam a escolher o computador em função de seus recursos sonoros. Os micros mais sofisticados possuem geradores de som de vários canais, embutidos, e permitem a programação dos mesmos através de comandos em BASIC.

Além de compor e executar músicas em um micro, o usuário tem, agora, a opção de conectar sua máquina a instrumentos musicais eletrônicos, como sintetizadores, órgãos e outros. Um padrão de interface, chamado MIDI (*Musical Instrument Digital Interface*), criado há poucos anos por um consórcio de indústrias eletrônicas japonesas e européias, está se firmando rapidamente, e promete abrir novas e numerosas possibilidades para o uso do computador em música. Examinaremos aqui esse periférico e suas características.

PRODUÇÃO DE SOM

A técnica de geração de efeitos sonoros e musicais em um micro evoluiu muitíssimo em relação aos "bipes" emitidos pelas primeiras marcas surgidas no mercado. Dos computadores cobertos por *INPUT*, o TRS-80 e o Apple são os que oferecem menos recursos para a geração de sons, exigindo programação elaborada em linguagem de máquina para a produção de algo mais complexo. O TK-2000 e o Spectrum têm comandos em BASIC, como o *SOUND*, que facilitam bastante a programação de efeitos sonoros, mas ainda em um nível mais simples. Já o TRS-Color e o MSX dispõem de recursos bem sofisticados, via comando **PLAY**. O MSX é o primeiro micro nacional com propriedades de sintetizador, pois permite o controle individual de algumas características da onda sonora, como a envoltória. Isso é possível graças a um circuito integrado específico para o controle de som.

Se você experimentou os programas para composição e execução de músicas, dados em artigos anteriores (como os das páginas 741 e 1009), certamente já tem uma idéia do que o seu computador pode (ou não) fazer. Seja através de recursos sonoros já embutidos na configuração básica do micro, seja através da adição de interfaces especiais (das quais existe uma grande variedade no mercado), as características mais importantes a observar em um bom sistema de geração sonora são:

- número de canais de saída sonora, ou vozes. Com apenas um canal, a música gerada é monofônica; o ideal é dispor de um mínimo de três vozes, para a sintetização de três instrumentos tocando simultaneamente;

- velocidade: se for muito baixa, não permite a execução de acordes múltiplos em cadência rápida;

- controle individual tanto de timbre como de intensidade;

- controle completo das características da onda sonora: bordo de ataque, bordo de fuga, envoltória etc. Esses recursos permitem sintetizar qualquer tipo de instrumento musical existente, ou até mesmo criar vozes para instrumentos que não existem;

- facilidade de programação: de preferência, deve ser possível utilizar comandos em BASIC;

- processador independente: permite que a melodia se inicie no ponto determinado pelo software principal, sendo executada independentemente do que a UCP estiver fazendo.

Apesar de toda a sofisticação das interfaces musicais de última geração, é preciso ter sempre em mente que a música gerada por um microcomputador nunca atinge os padrões de desempenho da música instrumental.

Devemos lembrar, ainda, que o teclado de um micro não corresponde ao teclado de um instrumento musical, como o piano, por exemplo, e é, na verdade, muito desajeitado para uso mais "sério". É aqui que entram em cena os instrumentos musicais digitais.

INSTRUMENTOS MUSICAIS

O desenvolvimento de novos instrumentos musicais nas últimas décadas é marcado por passagens semelhantes às que se observam na história das máquinas de calcular. Tradicionalmente, todos os instrumentos eram mecânicos: compunham-se de cordas, membranas, tubos etc. Aos poucos, a necessidade de obter intensidades sonoras maiores em *shows* destinados a grandes públicos (início da era do *rock*), assim como a intensa demanda criada pelas gravações, levou à eletrificação (ampliação eletrônica) de vários instrumentos musicais, como guitarras, órgãos e baterias. Finalmente, começaram a surgir instrumentos puramente eletrônicos — primeiro analógicos, e depois digitais — como o sintetizador.

As máquinas de calcular, por sua vez, evoluíram do ábaco para as calculadoras mecânicas movidas a manivela ou a motor elétrico, que, posteriormente, foram substituídas pelas modernas calculadoras eletrônicas. Como estas, os instrumentos musicais atuais estão recheados de *chips* integrados.

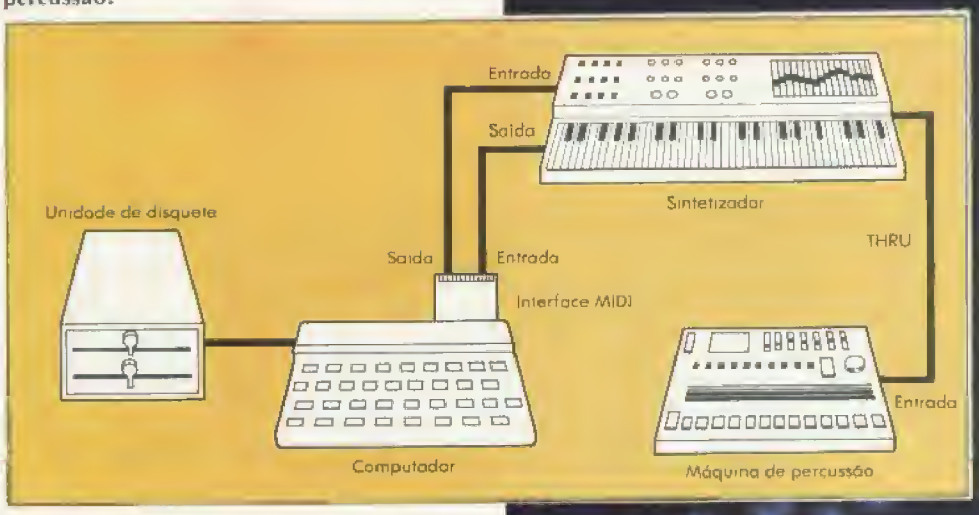
Os sintetizadores digitais são dispositivos sofisticadíssimos. Não se limitam à gama de notas e efeitos especiais oferecidos pelos micros (mesmo os mais modernos): dispõem de uma espantosa parafernália de recursos.

Um sintetizador comum, na faixa média de preço, permite a execução de acordes de até oito notas em um teclado como o de um piano. Quase todas as máquinas contam com um conjunto considerável de sons, envoltórias, timbres e ritmos especiais, possibilitando ao músico escolher instantaneamente entre um piano com eco, um violino ou um oboé, com acompanhamento de valsa, *bebop* ou batuque, em duas ou três vozes, com ou sem percussão.

Sons e instrumentos totalmente diferentes ou mesmo bizarros, efeitos de sirenas, explosões, gargalhadas, periquitos — ou o que vier à cabeça do executante — estão entre os recursos dos modelos de maior preço. Por meio de algumas teclas, eles nos dão acesso a uma orquestra completa.

- SINTETIZADORES
- INSTRUMENTOS MUSICAIS
- TECLADO
- MÁQUINAS DE PERCUSSÃO
- A INTERFACE MIDI
- CONEXÃO DO MICRO AO SINTETIZADOR
- POSSIBILIDADES E APLICAÇÕES
- SOFTWARE

Uma rede MIDI ligando um computador, um sintetizador e uma máquina de percussão.



Em geral, o tipo mais comum de sintetizador utiliza o teclado de piano ou órgão como dispositivo de execução. Mas como o sintetizador é, na realidade, uma caixa cheia de circuitos eletrônicos, capazes de receber sinais das mais diversas origens, podemos recorrer a qualquer tipo de instrumento musical para gerá-los. Os mais utilizados, atualmente, são os de cordas (guitarra, contrabaixo etc) e os de percussão (bateria eletrônica).

Tomemos como exemplo as máquinas de percussão: elas podem funcionar isoladamente ou embutidas dentro de um sintetizador. A qualidade artificial da percussão eletrônica, evidente nos primeiros modelos, tornou-se praticamente imperceptível nos sintetizadores modernos. Muitos deles têm memória RAM que permite o armazenamento de seqüências complexas ou não rítmicas, para execução posterior.

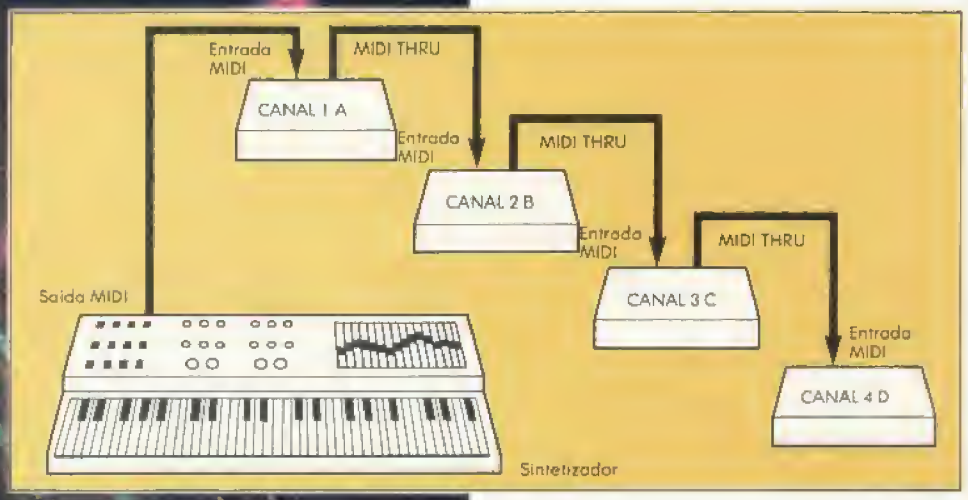
Essa capacidade dos sintetizadores está promovendo uma verdadeira revolução na maneira de se fazer música. Até recentemente, a habilidade manual — a capacidade de mover os dedos com desenvoltura e rapidez sobre as cordas ou teclas ou de fazer soar um acorde ou batida no momento exato — era indispensável à execução musical. Com o advento do instrumento musical programável, isso está mudando.

O instrumento programável age no sentido de "liberar" o talento musical de uma dependência da destreza manual. Mas, com certeza, não o substitui: a sensibilidade para o ritmo e a capacidade de compor músicas mentalmente ou de obter um efeito desejado sempre serão necessárias.

O sintetizador musical pode gerar sons de vários intrumentos musicais através do teclado. Se, por outro lado, você tiver a capacidade de programar e armazenar seqüências complexas de melodias, e tocá-las à vontade, até simultaneamente, seu "gênio musical" será enormemente ampliado.

Mas, antes que você se entusiasme, convém saber que os sintetizadores musicais programáveis são muito caros. E

Cada canal de informação de uma rede MIDI controla um instrumento.



O QUE É MIDI?

MIDI é um padrão industrial — da mesma forma que uma interface serial RS-232 ou uma paralela Centronics, usadas para conectar periféricos a um computador. Embora dirigida exclusivamente ao mundo da música, a MIDI desempenha um papel semelhante ao dessas interfaces, proporcionando um protocolo padronizado de transferência de informação entre o computador e um tipo especial de periférico: o instrumento musical digital. Isso assegura que qualquer instrumento (que seja compatível com o padrão MIDI) "entenda" o que o computador está ordenando.

Já se registraram outras tentativas de desenvolver um padrão de comunicação entre instrumentos musicais e computadores, mas nenhuma delas obteve uma grande aceitação. A MIDI, ao contrário, parece determinada ao sucesso: pelo menos os dois maiores fabricantes mundiais de sintetizadores e máquinas de percussão (Roland e Yamaha) já aderiram ao padrão MIDI. Além disso, vários microcomputadores lançados nos últimos anos, a começar do MSX, possibilitam a conexão à MIDI sem maiores dificuldades. Em conseqüência, prevê-se que esse padrão será universalmente adotado em pouco tempo.

Cada peça de equipamento compatível com a MIDI tem três soquetes de cinco pinos, do tipo DIN. Eles são rotulados de IN, OUT e THRU (alguns equipamentos MIDI mais antigos podem não ter este último tipo de conector). O soquete IN permite a recepção de sinais gerados em outro equipamento MIDI. THRU envia uma cópia idêntica dos sinais de entrada de um MIDI para ou-

é exatamente aqui que entra em cena a nova interface MIDI, salvando da frustração o amante da música que não tem um alto poder aquisitivo.

Com os computadores e os instrumentos musicais modernos utilizando a mesma tecnologia básica, tornou-se fácil interconectá-los e transmitir informações de um para o outro. É exatamente para isso que serve a MIDI.

tro, possibilitando a ligação de vários equipamentos "em cascata" (um equipamento que não tiver o soquete THRU é mais limitado, pois só pode ser conectado ao final de uma cadeia). OUT, finalmente, permite enviar sinais gerados em um equipamento para outro.

O padrão MIDI comporta até dezesseis canais simultâneos (paralelos) de informação. Cada canal controla um instrumento separado, mas a informação de vários instrumentos coexiste nos mesmos condutores elétricos (um artifício técnico denominado *multiplexação de alta velocidade*). Os instrumentos se encarregam de "sintonizar" a informação a eles encaminhada, da mesma maneira que um aparelho de televisão seleciona diversos canais.

UTILIZAÇÃO

Embora exista desde 1982, so recentemente a MIDI recebeu mais atenção do público. No Brasil, o interesse por essa interface surgiu com o advento dos primeiros microcomputadores compatíveis com a linha MSX. Entre suas aplicações mais freqüentes, destaca-se o comando de um instrumento musical por outro. Interconectando um sintetizador e um órgão eletrônico, por exemplo, a MIDI permite que ambos sejam comandados de um único teclado.

A MIDI também possibilita a sincronização de dois instrumentos diferentes — como uma máquina de percussão e um piano elétrico —, assim como a ligação de um *seqüenciador*. Esse dispositivo "memoriza" o que foi tocado na ordem correta, podendo repetir a execução de forma idêntica, tanto em tempo real quanto passo a passo. No primeiro caso, o seqüenciador reproduz exatamente o que o músico tocou; já no segundo, ele executa, etapa por etapa, trechos da melodia, permitindo que o músico acrescente outras notas, de modo a preencher os segmentos de tempo.

MIDI E COMPUTADORES PESSOAIS

Como já mencionamos, o "casamento" entre computadores pessoais e a interface MIDI veio a público com o lançamento da linha MSX. Os fabricantes japoneses da linha MSX tinham um grande interesse nessa associação, uma vez que a maioria deles opera também com divisões altamente rentáveis de instrumentos musicais digitais. A Yamaha introduziu, com um custo muito baixo, um sintetizador semiprofissional completo: o modelo CX5M, que é um micro MSX com um sintetizador embutido e um teclado de piano. Essa máquina abre uma série de possibilidades interessantes para o músico, como a exibição na tela de vídeo das notações musicais de uma composição ou o uso do computador como um seqüenciador de alta capacidade de memória, sem a necessidade de equipamentos extras.

Modelos diversos de microcomputadores — da linha MSX ou de outras linhas — podem ser utilizados com um equipamento MIDI acrescentando-se uma interface especial, ligada ao conector externo de expansão.

É provável que o preço de um computador completo ainda seja mais baixo que o de um sintetizador MIDI compatível. Mas, da mesma forma que aconteceu com outros periféricos, inicialmente muito caros — como impressoras, disquetes e monitores a cores —, o preço dos sintetizadores deverá diminuir bastante em um futuro próximo.

Antes mesmo que os preços comecem a cair, os usuários atraídos pela combinação microcomputador-instrumento musical devem ser informados sobre algumas peculiaridades desse periférico. Convém ter claro, em primeiro lugar, que a capacidade de geração sonora própria do micro não é usada quando se trabalha com a interface MIDI: o som é sempre gerado pelo sintetizador ou outro instrumento digital externo ao computador. Assim, não há vantagem em comprar um micro mais caro, dotado dos últimos recursos em matéria de geração musical. Mesmo a memória extra, disponível nos micros profissionais, não é tão importante, pois os computadores pessoais têm memória mais do que suficiente para o desempenho da função de seqüenciador.

Vale a pena observar, também, que a qualidade de som disponível não é limitada pelo meio de registro utilizado. Como o som é armazenado de forma digital (imune a ruídos), tanto faz usar uma fita cassete ou um disquete — a qualidade será sempre comparável à de um Compact Disc (CD). Em outras palavras, obtém-se na saída exatamente o que se colocou na entrada.

SOFTWARE

Uma vez que ligamos o computador a um ou mais equipamentos MIDI, é necessário um software especial para acionar o conjunto. A variedade de software é ainda restrita, e os preços são mais altos do que a média. A situação tenderá a mudar à medida que o uso da MIDI se popularizar.

Apesar das limitações, funções sofisticadas — como seqüenciamento, emulação de estúdios multicanais, composição e edição de melodias — são possibilitadas pelo software.

Mesmo quem não é capaz de tocar uma só nota em um instrumento musical normal, terá facilidade em compor temas musicais complexos no computador e enviá-los, em seguida, para execução. A composição pode ser armazenada digitalmente em fita ou disco, executada novamente, modificada etc. Já existem cartuchos de EPROM para micros compatíveis com o padrão MIDI, com conjuntos de músicas prontas, para autoacompanhamento ou, simplesmente, para se tocar no computador, usando-o como um sistema de alta-fidelidade.

Um pacote de software típico para a linha MIDI é o compositor musical, cujo funcionamento é exatamente igual ao de um editor de textos, só que trabalhando com a notação musical convencional, desenhada sobre a pauta. As suas funções mais comuns são inserção de músicas, apagamento, edição, listagem em vídeo ou impressora e execução musical em diversas cadências.

Um bom software coloca ainda à disposição do usuário uma série de funções de controle do sintetizador, tais como o comando independente de várias vozes (um sintetizador polifônico de qualidade deve ser capaz de tocar até dezesseis notas simultaneamente), seleção de ritmos, timbres ou seqüências, mixagem etc. Se entre as características do sintetizador se incluir a capacidade de dividir o teclado, você poderá tocar um instrumento com a mão esquerda e outro com a direita.

PADRONIZAÇÃO

É possível enviar três tipos de informação através de uma interface MIDI: notas, mudanças de programa e mixagem de timbre. Atualmente, existe um conjunto padronizado de códigos MIDI que funciona como qualquer sintetizador compatível. Entretanto, esses códigos permitem apenas o controle das funções mais elementares. Funções especiais são controladas por seqüência de códigos, que usualmente variam de instrumento para instrumento. Como resultado, um programa feito para um sintetizador MIDI pode não funcionar corretamente com outro. Além disso, para se programar orquestrações complicadas, é preciso estar familiarizado com a enorme gama de alternativas de controle. Provavelmente, também essas dificuldades serão atenuadas no futuro.

COMPUTADORES QUE OUVEM

- UNIDADES DE RECONHECIMENTO
- TIPOS DE FALA
- TIPOS DE SISTEMA
- FUNCIONAMENTO E USOS

Como já vimos, os periféricos de síntese de voz capacitam o micro a falar. O que talvez você não saiba é que as máquinas também podem entender a fala humana. Veja como.

No artigo *Computadores que Falam* (página 446), examinamos o funcionamento e as aplicações dos periféricos de síntese vocal, já disponíveis para a maioria dos microcomputadores. Como observamos, eles não são, em termos técnicos, muito complexos — existem modelos de preço acessível até para micros domésticos pequenos.

Se conferíssemos ao computador a habilidade inversa — ou seja, entender aquilo que falamos —, teríamos o "computador do futuro", que não precisaria de vídeo, nem de teclado para "conversar" conosco, humanos.

Mas isso não é tão fácil. Dotar a máquina da capacidade de reconhecer a fala é uma tarefa bem mais complicada do que fazê-la gerar a fala. Por essa razão, ainda não há sistemas capazes de um reconhecimento total da fala, nem mesmo nos mais sofisticados computadores de grande porte.

TIPOS DE SISTEMA

Não queremos dizer, com isso, que não se tenham desenvolvido sistemas com uma certa capacidade de reconhecimento da fala. Esses sistemas existem, inclusive em versões para microcomputadores pessoais, desde 1980/81.

Devemos identificar, portanto, o grau de *desempenho* de uma unidade de reconhecimento automático da fala (RAF). Dois critérios são utilizados. No primeiro deles, os sistemas são classificados em: sistemas de *reconhecimento da fala contínua* e sistemas de *reconhecimento de elocuções isoladas*.

Um sistema de reconhecimento da fala contínua deveria ser capaz de "entender" 90% das palavras emitidas no discurso natural — por exemplo, o pronunciamento de um deputado, uma conversa telefônica etc. E isso deveria ser feito em tempo real, ou seja, simultaneamente à fala. Sistemas com esse grau de desempenho ainda não existem. Os poucos sistemas de reconhecimento da fala contínua desenvolvidos até agora funcionam apenas em computadores imensos e custam muito caro. Além disso, eles são capazes de reconhecer um número muito restrito de palavras (o vocabulário é pequeno).

Já os sistemas de reconhecimento isolado, fabricados em uma grande variedade de modelos, são bem mais simples e baratos. Eles podem reconhecer com grande precisão um número restrito de palavras ou frases, enunciadas isoladamente e com clareza. O tamanho do vocabulário varia entre doze e trezentas palavras ou frases curtas, dependendo da sofisticação do sistema.

O segundo critério utilizado para a classificação dos sistemas RAF diz respeito à dependência do locutor.

Os sistemas *independentes do locutor*, como diz o nome, são capazes de reconhecer a fala de qualquer pessoa, em qualquer situação. Tais sistemas são tão raros quanto os de reconhecimento da fala contínua. Mais comuns são os sistemas cujo desempenho depende da pessoa que fala. Nesse caso, para que o sistema seja capaz de reconhecer a fala com um mínimo de precisão, a pessoa que vai usá-lo precisa "treinar" o computador, repetindo cada palavra ou frase do vocabulário um certo número de vezes. O programa analisa estatisticamente as repetições e as armazena na memória, para utilizá-las durante o processo de reconhecimento.

COMO FUNCIONA

O reconhecimento da fala requer o emprego de diversos algoritmos e processos. Inicialmente, a fala, captada por um microfone, é enviada ao computador e digitalizada por um conversor analógico digital. O conversor executa um processo de *amostragem*, ou seja, converte as ondas contínuas da fala em uma seqüência de números digitais a cada 50/100 milissegundos.

Os dados, armazenados em um *buffer* (memória intermediária), são processados por um saftware especial, que analisa o conteúdo de freqüência de cada amostra (análise espectral). Isso resulta em uma matriz (tabela), em que cada coluna corresponde a uma amostra, e cada linha, a uma freqüência. A matriz referente à palavra a ser identificada é comparada com as matrizes-padrão armazenadas internamente, uma para cada palavra do vocabulário. A palavra que, na comparação, acusar maior incidência é a palavra mais provável.

INTERFACES PARA MICROS

Existem, no exterior, diversas interfaces — em versões destinadas a microcomputadores pessoais ou profissionais — para reconhecimento automático da fala. Todos os modelos atualmente disponíveis empregam sistemas de reconhecimento de elocuções isoladas, dependente do locutor.

Para as linhas Apple, TRS-80 e TRS-Color, por exemplo, os sistemas Dragon e VoiceBox permitem o reconhecimento de 32 a trezentas palavras ou frases curtas, com 80% ou mais de precisão. Compõem-se de uma placa ou caixa de expansão, contendo um conversor AD, um conjunto de circuitos integrados especializados e software — parte do qual residente na memória ROM da unidade, parte em disquete.

Para utilizar o sistema, geralmente conecta-se o mesmo a uma porta de entrada serial do computador (RS-232C). Após carregar o software residente em disquete, o sistema pode ser operado com o auxílio de um microfone.

APLICAÇÕES

As aplicações do reconhecimento automático da fala são extremamente variadas, incluindo, por exemplo, o ensino de línguas, auxílio a pessoas inválidas, controle de aparelhos domésticos pela voz e até programação (existe uma versão "falada" do BASIC).

Existem também jogos muito divertidos para micros, como um jogo de pôquer, que utilizam voz sintética e reconhecimento da fala.

OS SEGREDOS DO TRS-80 (4)

Você sabe converter o sistema de posições de vídeo do seu micro em um sistema de coordenadas do tipo X e Y? Neste artigo, ensinaremos este e outros truques aos usuários do TRS-80.

Prosseguindo nossos estudos sobre a organização da tela dos micros compatíveis com a linha TRS-80, examinaremos como transformar o sistema de posições de vídeo, baseado na notação **PRINT @**, em um sistema de coordenadas verdadeiras, X e Y — recurso disponível para os usuários das linhas MSX, Apple, TK-2000 e Sinclair.

Relembrando o que já foi apresentado nesta série de artigos, a tela do TRS-80 é organizada em 1024 posições seqüenciais, numeradas de 0 (canto superior esquerdo da tela) a 1023 (canto inferior direito). A numeração aumenta da esquerda para a direita em uma linha. O número-índice de cada posição na tela é o utilizado na expressão @ (arroba) de um **PRINT** posicional.

POSICIONAMENTO RELATIVO

A expressão @ corresponde a um posicionamento absoluto, ou seja, indica o local exato da tela onde deve começar uma impressão. Em diversas aplicações, porém, é interessante realizar posicionamentos relativos — como avançar para a próxima posição na linha de baixo, recuar uma posição na mesma linha, ir para o final da mesma linha e assim por diante.

Por meio de algumas expressões matemáticas, podemos realizar facilmente o posicionamento relativo, de modo a tratar a tela como um plano bidimensional. Essas expressões seguem um formato comum, em que se executa o cálculo de conversão e, ao mesmo tempo, se impõem os limites de variação mínima e máxima. Por exemplo: se a posição atual do cursor é **P** (um número inteiro entre 0 e 1023), podemos fazer com que ele avance uma posição para a direita por meio da expressão:

```
P%=P%-(P%+1<1024)
```

A notação intrínseca P% é utilizada para indicar que **P** é um número inteiro. Evitamos, assim, alguns erros de arredondamento nas expressões. A expressão acima combina, em uma única fórmula, uma adição e um teste:

```
P%=P%+1:IF P%>1023 THEN P%=1023
```

Ou seja, a posição P% sofre um incremento de 1, ao mesmo tempo em que é forçada a não exceder 1023.

Analisando a expressão completa, vemos que a subexpressão entre parênteses (P% + 1 < 1024) terá um valor igual a −1 se for verdadeira, isto é, se a próxima posição P% for menor que 1024. Nesse caso, ela será reduzida a P% = P% + 1, causando um incremento. Se a subexpressão for falsa, isto é, se P% + 1 for igual a 1024, seu valor será igual a 0 e P% receberá um incremento de 0 (ou seja, não será incrementado).

Em seguida, listamos as expressões mais úteis para posicionamento relativo. Estude-as cuidadosamente para entender como funcionam.

CONVERSÃO PARA COORDENADAS

O endereçamento linear da tela, tal qual é usado pelo interpretador BASIC do TRS-80, apresenta uma série de inconvenientes para a impressão de desenhos e textos que requerem o emprego de um sistema de coordenadas ortogonais (semelhante ao existente para a tela gráfica, com os comandos **SET**, **RESET** e **POINT**). Entretanto, não é difícil especificar um conjunto de equações que promova a transformação de um sistema de referência em outro.

Em primeiro lugar, precisamos adotar uma convenção referente à origem do sistema de coordenadas ortogonais. Se a origem deste corresponder ao canto superior esquerdo (X = 0 e Y = 0), e se X indicar a coluna, e Y, a linha da tela, estaremos usando a mesma convenção estipulada para os gráficos com **SET**. Neste caso, teremos:

1. Converter (X,Y) para uma posição @:

```
P% = 64*Y + X
```

2. Converter uma posição @ para (X,Y):

```
X = P%-INT(P%/64)*64
Y = INT(P%/64)
```

Se quisermos adotar a convenção cartesiana clássica (origem no canto inferior esquerdo), teremos:

1. Converter (X,Y) para uma posição @:

```
P% = 64*(15-Y) + X
```

2. Converter uma posição @ para (X,Y):

```
X = P%-INT(P%/64)*64
Y = 15-INT(P%/64)
```

O TECLADO DO TRS-80

O teclado dos micros compatíveis com a linha TRS-80 apresenta certas particularidades que, uma vez conhecidas, permitem a programação de alguns truques muito interessantes.

Da mesma forma que o vídeo, o teclado do TRS-80 tem uma área de memória, na RAM, reservada à armazenagem dos dados da matriz de entrada. Cada tecla, quando pressionada, envia um sinal por meio de dois condutores: um horizontal (percorre as fileiras de teclas) e outro vertical (percorre as colunas). Isso corresponde a um verdadeiro endereço da tecla, que tem seu correspondente em uma unidade da RAM.

Para compreender de que maneira o

MICRO DICAS

PROTEJA O SEU PROGRAMA

Utilize o que você aprendeu sobre a desativação da tecla <BREAK> no TRS-80 para proteger o seu precioso programa em BASIC contra a curiosidade dos "piratas" de software.

A operação é simples: faça o computador carregar e executar automaticamente o programa. A primeira linha executável deve conter os comandos **POKE**, vistos neste artigo, que desativam a resposta à tecla <BREAK>. Essa providência impedirá que o programa seja interrompido e listado por meio do comando **LIST**.

Se o seu TRS-80 tiver o sistema operacional de disquetes (DOS), coloque o comando **BASIC NOME** (nome dado ao programa) no arquivo de auto-execução. Mas lembre-se de gravar NOME usando a opção de proteção contra listagem (**SAVE "NOME", P**).

- A NOTAÇÃO @
- POSICIONAMENTO RELATIVO
- LIMITES DA TELA
- FÓRMULAS DE CONVERSÃO
- COORDENADAS ORTOGONAIS
- O TECLADO DO TRS-80
- BLOCO DE CONTROLE
- POSIÇÃO DO CURSOR
- TECLAS AUTO-REPETITIVAS
- A TECLA <**BREAK**>

teclado é processado e controlado pelo BASIC, precisamos conhecer uma outra área de memória RAM, que abrange diversos endereços — chamados coletivamente de *bloco de controle* do teclado. Como veremos em seguida, muitos desses endereços são úteis para nossos truques de programação.

A POSIÇÃO DO CURSOR

O BASIC do TRS-80 tem uma função intrínseca, chamada **POS(0)**, que informa a posição corrente do cursor dentro de uma linha. O número fornecido por essa função varia entre 0 e 63, e não informa a posição bidimensional na tela (posição @), que corresponde a um número de 0 a 1023.

A posição do cursor é armazenada no bloco de controle do teclado, nos bytes 16416 e 16417 (é um número de dezesseis bits, portanto). Para determinar a posição do cursor, basta usar um par de comandos **POKE**:

```
V = 256*PEEK(16417)+PEEK(16416)
```

Por meio dessa expressão, obtemos diretamente o endereço absoluto da memória de vídeo onde está o cursor. Para converter o valor resultante em uma posição **PRINT @**, basta subtrair o valor 15360, que corresponde ao primeiro endereço da memória de vídeo:

```
P = 256*PEEK(16417) + PEEK
(16416)-15360
```

Rodando o próximo programa, você terá uma demonstração desse cálculo. O programa preenche a tela com números inteiros, até chegar à posição 960, onde é chamada uma rotina de interrupção. Pressionando-se qualquer tecla, a tela é limpa, e a impressão continua a partir do último número exibido.

```
100 DEFINT N:N=0
110 CLS
120 N=N+1:PRINT N
130 NP=256*PEEK(16417)+PEEK(
16416) - 15360
140 IF NP<960 THEN 120
150 PRINT "APERTE <ENTER> P/
CONTINUAR"
160 IF INKEY$="" THEN 160 ELSE
110
```

REPETIÇÃO AUTOMÁTICA DE TECLAS

O teclado original do TRS-80, assim como o de seus compatíveis, não é repetitivo — ou seja, nada acontece se mantemos pressionada uma tecla. Trata-se de uma deficiência, pois seria conveniente, em muitas aplicações (jogos, por exemplo), que algumas teclas fossem auto-repetitivas.

Mas existe uma solução para esse problema: podemos saber se uma tecla está sendo pressionada ou não por intermédio do endereço 14591 do bloco de controle do teclado, que contém essa informação. Basta, portanto, consultar uma vez esse endereço — a cada repetição de um laço, por exemplo — e direcionar a lógica do programa de acordo com a informação obtida.

Para verificar a operacionalidade do endereço 14591 em seu micro, digite o seguinte programa:

```
10 PRINT PEEK(14591);:GOTO 10
```

Em seguida, digite **RUN** e experimente pressionar qualquer tecla. Note que, enquanto não se pressiona nenhuma tecla, o programa fica imprimindo zeros na tela. Um número maior que zero aparece quando se pressiona alguma tecla ou combinação de teclas.

O próximo programa ilustra com bastante clareza essa aplicação: uma espécie de "minhoquinha" é formada sobre a tela, quando se pressionam as quatro teclas com as flechas:

```
100 DEFINT A-Z:X=64:Y=24
110 T$ = CHR$(9) + CHR$(8) +
CHR$(91) + CHR$(10)
120 CLS
130 SET(X,Y):IF PEEK(14591)>0
THEN 160
140 C$=INKEY$:IF A$="" THEN 130
150 C=ASC(C$):GOTO 130
160 ON INSTR(T$,A$) GOSUB 200,
250, 300, 350
170 GOTO 130
200 X=X+1:IF X>127 THEN X=127
210 RETURN
250 X=X-1:IF X<1 THEN X=1
260 RETURN
300 Y=Y-1:IF Y<1 THEN Y=1
310 RETURN
400 Y=Y+1:IF Y>47 THEN Y=47
410 RETURN
```

O que é edição em tela completa?

Em um banco de dados, a ficha de entrada de informações é exibida na tela com todos os campos. *Edição em tela completa* é o recurso que permite ao usuário "passear" com o cursor por toda a tela, corrigindo e mudando as informações contidas nos vários campos, até ficar satisfeito com o resultado. Em seguida, por meio de uma tecla de controle, o conteúdo é armazenado na memória.

Conhecendo as funções do vídeo e do teclado do TRS-80, você não terá dificuldade em desenvolver um programa de edição em tela completa. Observe a seguinte estrutura:

- Inicialmente, a tela é limpa e os campos são exibidos na tela. Use comandos **PRINT @** para isso;
- em seguida, a sub-rotina de entrada de dados é chamada por **INKEY$**. Coloque nela vários **IF**, para detectar se alguma tecla de controle foi pressionada. Localize o cursor e efetue a operação solicitada;
- finalmente, tendo pressionado a tecla que assinala o fim da edição, leia (com **PEEK**) o conteúdo dos campos e transfira-o para a memória.

BLOQUEIO DA TECLA <BREAK>

Podemos desativar a tecla <**BREAK**> do TRS-80 Modelo 1 ou Modelo 3 (e dos microcomputadores compatíveis) se dermos os seguintes comandos (em modo direto, ou dentro de um programa):

```
POKE 16396,175:POKE 16397,201
```

Para ativar novamente, digite:

```
POKE 16396,201
```

Esse recurso é de grande utilidade quando desejamos impedir que um programa seja interrompido — intencional ou acidentalmente — pelo usuário.

PROGRAMANDO EM LOGO

A linguagem LOGO foi desenvolvida para facilitar o aprendizado da programação. Mas, se você pensa que ela se destina apenas a crianças, ficará surpreso com a riqueza de recursos.

Os computadores da década de 60 eram extremamente caros. A potência e a capacidade hoje disponíveis em um simples micro doméstico custavam alguns milhões de dólares nos computadores de grande porte de então, pois mesmo as máquinas maiores e mais sofisticadas contavam com um máximo de 144 Kbytes de memória RAM.

Assim, por razões de economia, as primeiras linguagens de programação foram projetadas para ocupar a menor quantidade de memória possível. Priorizava-se a facilidade de sua implementação no computador, e não a facilidade de uso para o programador.

Com o aparecimento dos microcomputadores, por volta de 1975, as linguagens dos antigos computadores tornaram-se bastante populares entre os usuários dos micros, pois essas máquinas, no começo, também dispunham de uma memória muito limitada. A maioria das pessoas aceitava como natural as dificuldades de aprendizado do BASIC e outras linguagens do gênero. "O que é simples para o computador também é simples para o programador", dizia-se.

A ESCOLHA DA LINGUAGEM

Quase todos os micros operam, originalmente, com um interpretador BASIC residente na memória ROM. E esta é a única linguagem utilizada pela maioria dos proprietários de micro.

Entretanto, não há motivo para que se limitem a uma única linguagem. O BASIC é apenas um programa em código de máquina, que pode tanto residir na memória quanto ser carregado de uma fita ou disquete. Nos micros profissionais, por exemplo, o BASIC é uma linguagem a mais, e precisa ser carregada quando se quer usá-la em programação. Isso significa que é perfeitamente possível carregar interpretadores de outras linguagens — ou seja, podemos mudar a linguagem que o computador "entende". Esses interpretadores são programas em código de máquina, capazes de entender o vocabulário e a sintaxe de uma linguagem de programação, e de traduzi-los em instruções executáveis pela UCP do micro.

No artigo da página 1288, vimos que, desde o aparecimento dos primeiros computadores, mais de duzentas diferentes linguagens de programação foram desenvolvidas. Essas linguagens servem aos mais variados propósitos. Muitas delas são extremamente especializadas, encontrando aplicação apenas em campos de pesquisa muito sofisticados. Outras são tão genéricas e fáceis de usar quanto o BASIC, e já estão disponíveis para microcomputadores.

Se você quiser utilizar uma determinada linguagem de programação no seu modelo de microcomputador, precisará, antes de mais nada, achar um programa interpretador ou compilador que possa ser executado nele. Linguagens alternativas são fornecidas em fitas ou discos, e devem ser carregadas na memória, antes de sua utilização. Algumas vezes, o interpretador também é encontrado em cartuchos removíveis da ROM (como os que existem para os micros TRS-Color e MSX), não precisando, evidentemente, ser carregado.

Os microcomputadores profissionais, com sistemas operacionais do tipo MS-DOS, CP/M ou UNIX, são os que dispõem de maior variedade de linguagens de programação. Já existe, porém, uma razoável oferta de linguagens alternativas para micros mais baratos — como os das linhas Spectrum, TRS-Color, Apple e MSX —, principalmente se puderem ser acoplados a disquetes.

NÍVEIS DE COMUNICAÇÃO

Uma linguagem de programação pode ser entendida tanto pelo usuário quanto pela máquina, constituindo uma espécie de mediação entre as linguagens de ambos — ou seja, entre a linguagem natural (o inglês, por exemplo) e a linguagem binária, empregada por todos os computadores existentes.

Quando a linguagem de programação está mais próxima da binária do que da natural, diz-se que é de *nível baixo*. O Assembler é um exemplo de linguagem desse tipo. Já as linguagens de *alto nível*, como o BASIC, são mais parecidas com a natural. As máquinas de quinta geração — a próxima geração de computadores — provavelmente irão utilizar linguagens superavançadas e poderão entender ordens dadas em linguagem natural ou seminatural. Mas as linguagens atuais — mesmo as mais avançadas, como o PROLOG — representam ainda um meio termo entre a facilidade de uso e a facilidade de implementação em um computador.

Além do BASIC, muito poucas linguagens de alto nível são utilizadas mais intensamente em microcomputadores pessoais: as principais são o LOGO e o PASCAL. Como veremos nos artigos desta série, essa linguagens são totalmente diferentes quanto a seu histórico e a seu campo de aplicação.

ORIGENS DO LOGO

Em 1967, um grupo de pesquisadores do famoso Massachusetts Institute of Technology (MIT), localizado na costa leste dos Estados Unidos, começou a explorar um caminho totalmente novo no desenvolvimento de linguagens para computadores. Esse grupo, liderado por Seymour Papert, um sul-africano expatriado, tinha como objetivo criar uma linguagem mais adaptada ao modo de funcionamento do intelecto humano do que ao processador central de um computador. Desse empenho nasceu o LOGO.

Papert tinha trabalhado com o famoso psicólogo e filósofo suíço Jean Piaget. As pesquisas de Piaget sobre o desenvolvimento da cognição evidenciavam que uma criança só é capaz de entender um conceito abstrato quando ele é apresentado por meio de exemplos concretos. As condições ideais de aprendizado estariam, assim, condicionadas ao envolvimento e experimentação pessoais. Essa abordagem influenciou fortemente a elaboração do LOGO.

Outra influência importante foi o trabalho de Marvin Minski, pesquisador do MIT e um dos pais da Inteligência Artificial — ciência que procura reproduzir aspectos da inteligência humana em computadores.

Os computadores não são inteligentes: sua capacidade se limita à obediência de instruções dadas com grande detalhe e precisão. A resolução de um pro-

- ESCOLHA E DISPONIBILIDADE DE LINGUAGENS
- O QUE É LOGO
- FUNDAMENTOS PSICOPEDAGÓGICOS
- A TARTARUGA
- PSEUDO-LOGOS
- UM LOGO PARA SEU COMPUTADOR
- PROGRAMAÇÃO

blema intelectual envolve fatores tão complexos e variados, que programar uma máquina para uma operação desse tipo constitui uma tarefa especialmente desafiadora. Minski e seus colaboradores dedicaram-se a ela, buscando desenvolver uma linguagem de programação que facilitasse a simulação da capacidade de decisão e de resolução de problemas. Chegaram ao LISP (*LISt Processing*) — linguagem que passaram a utilizar desde o início dos anos 60.

A estrutura básica de dados do LISP é a *lista*, uma coleção de objetos elementares (chamados de *átomos*). Uma lista pode fazer parte de outras listas, o que torna mais fácil representar e processar dados simbólicos (não numéricos). Entretanto, o LISP é muito difícil de aprender.

O LOGO é essencialmente um dialeto do LISP — contém todas as estruturas e comandos que essa linguagem emprega para representar e processar dados simbólicos (palavras, listas etc.). Além disso, possui poderosos comandos gráficos, incluídos para facilitar seu uso por crianças pequenas.

DEMONSTRAÇÕES CONCRETAS

Papert e seus colegas estavam interessados em uma linguagem que permitisse iniciar as crianças no mundo da programação. Três áreas foram privilegiadas: desenho, música e robótica. Isto resultou da constatação de que as crianças se sentem mais motivadas a usar o computador quando a tarefa a realizar consiste em desenhar na tela, tocar música ou movimentar pequenos robôs com o auxílio do computador. Este último interesse levou Papert a criar uma "tartaruga" robótica.

Na época, não se encontravam monitores gráficos de vídeo a preços acessíveis. A tartaruga, um pequeno robô móvel sobre rodas, supria essa deficiência: carregava uma caneta em sua "barriga", e, sob o controle de comandos específicos do LOGO, podia abaixá-la e levantá-la. Assim, era capaz de desenhar figuras no chão, obedecendo às ordens da criança, que passava a relacionar os comandos da linguagem com os movimentos da tartaruga e com conceitos de orientação e geometria.

Quando surgiram os computadores pessoais, porém, a tartaruga-robô passou a correr risco de "extinção", pois era muito mais simples e barato imitar seu movimento na tela por meio de gráficos. Sua descendente direta é a tartaruga gráfica bidimensional — um simples cursor, em geral de forma triangular, que obedece às mesmas instruções de deslocamento usadas para movimentar a tartaruga-robô. Mas o simpático "animalzinho" não chegou a desaparecer. Ao contrário, tem se observado uma tendência a trazê-lo de volta às salas de aula. O pequeno robô é, sem dúvida, bem mais adequado ao ensino de crianças pequenas, que não entendem o formalismo da tartaruga gráfica. Além disso, é muito mais divertido!

OS ELEMENTOS DA LINGUAGEM

O que tem a ver a movimentação de uma tartaruga de brinquedo com programação de computadores e psicologia infantil? Papert vê no computador um importante veículo para a criatividade e a expressão de idéias, e acredita que se deve ensinar as crianças a dominá-lo, evitando que sejam dominadas por ele. A melhor maneira para que uma criança aprendesse a manipular computadores seria viver em uma "cultura computacional", assim como a melhor maneira de aprender italiano é viver na Itália. O LOGO — dando à criança os meios para usar os poderosos recursos disponíveis no computador — é a sua proposta para realizar isso. Como Papert procura demonstrar em seu livro *LOGO, Computadores e Educação* (*Mindstorms, Children, Computers and Powerful Ideas*, no original), essa linguagem funciona não só como uma ferramenta para a resolução de problemas, mas como um verdadeiro sistema de referência — denominado *micromundo* — para a introdução de novas idéias e conhecimentos.

Logo após o aparecimento do primeiro interpretador LOGO para computadores de grande porte, surgiram as primeiras versões para micros. Por sua capacidade gráfica superior, a linha Apple foi privilegiada com as versões iniciais mais poderosas da linguagem. Posteriormente, outros micros — entre os quais o Spectrum, o MSX, e o TRS-Color — ganharam interpretadores LOGO compatíveis com o padrão MIT.

Como o campo de aplicação do LOGO é eminentemente educacional, os simbolismos da palavra escrita para a descrição da forma e funcionamento do nosso mundo tem grande importância. Assim, esta foi a primeira linguagem a ser amplamente traduzida para idiomas locais, inclusive o português. A partir de 1974, especialistas da Unicamp (Universidade Estadual de Campinas, no Estado de São Paulo) começaram a produzir o BRASLOGO, a versão brasileira do LOGO, que mais tarde foi adotada por diversas empresas fabricantes de microcomputadores, como a Itautec, a Gradiente e a Microdigital. Hoje, há versões dessa linguagem para as linhas Apple, MSX, TK-90X e CP-400, entre outras. Existem também diversos "pseudo-LOGO", assim chamados por serem versões muito reduzidas ou limitadas do LOGO original, do qual aproveitam apenas os recursos gráficos. Como o ColorLOGO, do TRS-Color 1, esses pseudo-LOGOS não contam com os recursos de processamento de listas, funções matemáticas etc. da versão completa.

LOGO foi a primeira linguagem simbólica "amistosa", ou seja, fácil de usar. Como ela se presta a aplicações educacionais, muitos julgam que se destina exclusivamente às crianças. Nada mais longe da verdade. Os recursos intrínsecos do LOGO (recursão verdadeira, processamento de listas etc.) colocam-no no mesmo *ranking* que o LISP, o SNOBOL, o PROLOG e outras linguagens empregadas em campos complexos, como Inteligência Artificial.

UM LOGO PARA SEU MICRO

A primeira providência a ser tomada quando se quer programar em LOGO é carregar o interpretador da linguagem. Cada linha de computador tem uma versão especial, pois ainda não existe uma padronização do LOGO.

O micro Spectrum dispõe de diversas versões do LOGO — e de pseudo-LOGO — em inglês. Estas, contudo, só podem ser obtidas no exterior.

No Brasil, a Microdigital lançou um LOGO em português para os computadores TK-90X e TK-95. Essa versão é carregada a partir de fita cassete e segue aproximadamente o padrão BRASLOGO (ou LOGO Itautec, desenvolvido pela Itautec em conjunto com a Universidade Estadual de Campinas).

Os modelos da linha TRS-Color só dispõem de um interpretador LOGO em cartucho ou disquete, com comandos e mensagens em inglês. Uma versão antiga, o ColorLOGO, é apenas um pseudo-LOGO gráfico, embora muito poderoso.

Esta é a linha que, internacionalmente, dispõe do maior número e variedade de interpretadores LOGO, todos eles carregados a partir de disquete. Em inglês, existem várias versões bastante eficazes, como o Terrapin LOGO, o Apple LOGO etc., que implementam o padrão MIT, além de terem comandos para geração de música, controle de tartaruga-robô e outros recursos.

Em português, a versão mais difundida é o MLOGO, desenvolvida por uma software-house paulista, a Micro-Arte. Essa versão, entretanto, não é compatível com o padrão BRASLOGO.

Os micros desta linha contam com vários interpretadores LOGO em inglês, em cartucho e em disquete. Todos aproveitam os excelentes recursos gráficos e sonoros do MSX para oferecer comandos especiais, que permitem o uso de sprites, tartarugas múltiplas etc.

No Brasil, há duas versões do LOGO em português. Uma delas, a da Gradiente, é apresentada em cartucho e seus recursos são bastante limitados. A outra — o HOTLOGO, da Sharp —, em cartucho e em disquete, é poderosíssima e segue o padrão BRASLOGO/Itautec.

As versões brasileiras aceitam palavras acentuadas segundo as regras da língua portuguesa.

PROGRAMANDO EM LOGO

Como não existe uma versão padronizada para o LOGO, todos os programas desta série serão dados em duas versões bem conhecidas: o padrão MIT, em inglês (listagens identificadas pelo símbolo do Apple), e o padrão BRASLOGO/Itautec, em português (listagens identificadas pelo símbolo do MSX).

No último artigo apresentaremos uma tabela de conversão de comandos para outras versões menos difundidas. Nas versões em português, a sintaxe é sempre a mesma; muda apenas a forma usada nas palavras-chave.

Quando se carrega o LOGO na memória (o que é desnecessário se a versão for em cartucho, pois, nesse caso, o programa se auto-executa), surge na tela uma mensagem inicial de "boas-vindas", que varia de acordo com o tipo de interpretador usado.

O sinal adotado pelo LOGO para avisar que está pronto a aceitar um comando é o ponto de interrogação. Ao lado desse sinal, aparece o cursor.

Inicialmente, vamos explorar os comandos gráficos do LOGO: colocaremos a tartaruga gráfica na tela e daremos comandos para movimentá-la, fazendo diversos desenhos. Para colocar a tartaruga na tela, digite este comando, e, depois, pressione a tecla <**ENTER**> ou <**RETURN**>:

```
SHOWTURTLE
```

```
TARTARUGA
```

O cursor aparece no centro da tela — sob a forma de um triângulo ou como uma tartaruga estilizada, conforme a versão do LOGO —, já pronto para desenhar. Por analogia com o robô mecânico, dizemos que a tartaruga está com a "caneta abaixada".

Dispomos de vários comandos de deslocamento da tartaruga. Ela pode andar para a frente ou para trás e virar para a direita ou para a esquerda. O deslocamento linear é medido em número de passos (*steps*) ou pontos gráficos, e a virada, em graus (ângulo). Por exemplo, para fazer a tartaruga andar trinta passos adiante, digite:

```
FORWARD 30
```

```
PARAFRENTE 30
```

Deixe um espaço entre a palavra de comando e o parâmetro (30, no caso). Sem esse espaço, o computador entenderá a linha como um comando único, que não é capaz de reconhecer, e mostrará uma mensagem de erro do tipo:

```
I DON'T KNOW HOW TO FORWARD30
```

```
VOCÊ AINDA NAO ME ENSINOU
PARAFRENTE30
```

À medida que a tartaruga se desloca, traça uma linha reta na direção desejada, pois, como dissemos, ela já aparece na tela em modo de escrita.

Uma linha de comando pode ser corrigida com as teclas do cursor, antes de se digitar <**ENTER**> ou <**RETURN**>.

Para alterar a direção em que a tartaruga se movimenta, usa-se o comando que indica a direção e o ângulo da virada. Experimente:

```
RIGHT 30
```

```
PARADIREITA 30
```

Com isso, o símbolo da tartaruga apenas se inclina na direção desejada. Para que ela avance, digite:

```
FORWARD 50
```

```
PARAFRENTE 50
```

A linha traçada pela tartaruga forma um ângulo de 30 graus à direita da linha vertical anterior.

Comandos sucessivos podem ser dados na mesma linha. Experimente:

```
LEFT 30 BACK 50
```

```
PARAESQUERDA 30 PARATRAS 50
```

A linha de comando ordena, em suma, que a tartaruga retorne à orientação original (isto é, "olhando" para a parte superior da tela), por meio de uma virada de 30 graus, e que ande para trás cinqüenta passos.

ABREVIATURAS

Para facilitar a digitação, podemos recorrer à forma abreviada de muitos comandos da linguagem LOGO. Veja abaixo as abreviaturas para os comandos que foram mostrados até agora:

```
SHOWTURTLE          ST
FORWARD             FD
BACK                BK
LEFT                LT
RIGHT               RT
```

```
TARTARUGA           TAT
PARAFRENTE          PF
PARATRAS            PT
PARAESQUERDA        PE
PARADIREITA         PD
```

MAIS COMANDOS

O LOGO possui uma grande variedade de comandos (também chamados *primitivos*). Vejamos mais alguns deles, relacionados à elaboração de gráficos:

HOME - Leva a tartaruga de volta ao centro da tela, sem apagá-la.

CLEARSCREEN - Apaga o conteúdo da tela e leva a tartaruga de volta ao centro (abreviação: **CS**).

PENUP - "Desliga" o modo de escrita da tartaruga: ao se movimentar, ela não deixará nenhum traço (abreviação: **PU**).

PENDOWN - "Liga" o modo de escrita da tartaruga (abreviação: **PD**).

PENCOLOR - Muda a cor do traço e das letras. É seguido de um número entre 0 e 6, que indica o código da cor (abreviação: **PC**).

Digite o programa que se segue para ter um exemplo da utilização dos comandos aprendidos até o momento. Nunca se esqueça de pressionar a tecla <**ENTER**> após cada linha de comando.

```
SHOWTURTLE
LEFT 45
FORWARD 71
RIGHT 135
PENUP
FORWARD 50
PENDOWN
LEFT 45
BACK 71
PENUP
HOME
PENDOWN
```

O comando **CLEARSCREEN** se encarregará de limpar toda a tela, caso você queira realizar outras experiências.

PARACENTRO - Leva a tartaruga de volta ao centro da tela, porém sem apagá-la (abreviação: **PC**).

APAGUEDESENHO - Apaga o conteúdo da tela e leva a tartaruga de volta ao centro (abreviação: **AD**).

USENADA - "Desliga" o modo de escrita da tartaruga: ao se mo-

vimentar, ela não deixará nenhum traço (abreviação: UN).

USELÁPIS - "Liga" o modo de escrita da tartaruga (abreviação: UL).

USEBORRACHA - Apaga todos os traços por onde passar a tartaruga (abreviação: **UB**).

MUDECL - Muda a cor do traço e das letras. Deve ser seguido de um número entre 0 e 7, que indica o código da cor.

Eis aqui um exemplo para ilustrar o uso dos comandos aprendidos até agora. Não se esqueça de pressionar a tecla <**ENTER**> após cada linha de comando.

```
TARTARUGA
PARAESQUERDA 45
PARAFRENTE 71
PARADIREITA 35
USENADA
PARAFRENTE 50
USELAPIS
PARAESQUERDA 45
PARATRAS 71
USENADA
PARACENTRO
USELAPIS
```

MONTE UM PROCEDIMENTO

Todos os comandos examinados foram dados em modo direto, ou "imediato", que equivale ao modo direto do BASIC — ou seja, o comando é executado imediatamente pelo interpretador, e logo esquecido. Entretanto, existe um outro modo em LOGO: o modo programado, ou modalidade procedimento.

Nessa modalidade, atribui-se um nome a um conjunto de comandos, que passam, então, a ser conhecidos como um *procedimento*. O nome que for dado ao procedimento torna-se parte integrante do vocabulário do LOGO. Dizse, por essa razão, que o LOGO pertence à família das linguagens extensíveis.

Quando se digita o nome de um procedimento registrado, o interpretador executa todos os comandos contidos no mesmo, em seqüência. Assim, pode-se dizer que o procedimento também é um programa. Mas, ao contrário de um programa em linguagem BASIC, dispensa a numeração das linhas.

Para iniciar a entrada de um novo procedimento, use o comando:

```
TO NOME
```

```
APRENDA NOME
```

Na versão em português, o comando **APRENDA** pode aparecer na sua forma abreviada, ou seja, **AP**.

O NOME do procedimento pode ser qualquer um e ter qualquer tamanho. Só não deve conter espaços em brancos ou coincidir com o nome dos primitivos da linguagem LOGO.

Você poderá entender melhor de que maneira funciona um procedimento por meio de um exemplo. Vamos definir um procedimento chamado ZIGZAG. Logo que digitamos o comando inicial — **TO ZIGZAG**, em inglês, ou **APRENDA ZIGZAG**, em português —, tudo o que estava na tela desaparece, e entra em ação um pequeno *editor de textos*, que permite a entrada dos comandos que constituirão o procedimento. O sinal de prontidão muda de ? (ponto de interrogação) para > (sinal de maior).

Para finalizar o procedimento, pressiona-se a tecla **END** (para as versões em inglês) ou **FIM** (versões em português). O sinal de ? aparece novamente, indicando que voltamos ao modo direto. Em alguns computadores, como o Apple, por exemplo, é necessário pressionar as teclas <**CTRL**> e <**C**>.

```
TO ZIGZAG
FORWARD 20 LEFT 150
FORWARD 20 RIGHT 150
FORWARD 20 LEFT 150
FORWARD 20 RIGHT 150
FORWARD 20 LEFT 150
FORWARD 20 RIGHT 150
END
```

```
APRENDA ZIGZAG
PARAFRENTE 20 PARAESQUERDA 150
PARAFRENTE 20 PARADIREITA 150
PARAFRENTE 20 PARAESQUERDA 150
PARAFRENTE 20 PARADIREITA 150
PARAFRENTE 20 PARAESQUERDA 150
PARAFRENTE 20 PARADIREITA 150
FIM
```

ZIGZAG é, agora, um novo comando do LOGO. Se você digitar esse comando, a tartaruga executará os comandos gráficos do programa anterior e traçará um ziguezague na tela.

Como você deve ter observado, o programa compõe-se de três repetições da mesma seqüência de comandos, uma para cada "perna" do ziguezague. Mas o LOGO tem um comando que permite especificar o número de repetições (semelhante ao **FOR ... NEXT** do BASIC). Utilizando-o, e abreviando os comandos gráficos, nosso programa ficará assim:

```
TO ZIGZAG
REPEAT 3 [FD 20 LT 150 FD 20 RT
150]
END
```

```
APRENDA ZIGZAG
REPITA 3 [PF 20 PE 150 PF 20 PD
150]
FIM
```

A rotina a ser repetida, delimitada pelos sinais [e] (não use parênteses), é precedida pelo comando de repetição e pelo número de vezes que ela deve ser repetida. O comando de repetição é muito útil para a obtenção de figuras geométricas regulares. Podemos traçar um semicírculo, por exemplo, com o seguinte comando:

MICRO DICAS

CONVERSÃO PARA O MLOGO

Um dos interpretadores LOGO mais usados no Brasil é o desenvolvido pela Micro-Arte, de São Paulo, para computadores da linha Apple. Como nessa versão os comandos primitivos do Apple LOGO foram traduzidos para o português em desacordo com o padrão BRASLOGO, apresentamos abaixo os comandos do MLOGO correspondentes aos utilizados neste artigo. A sintaxe permanece a mesma.

BRASLOGO	MLOGO
TARTARUGA	MOSTRET
PARACENTRO	CENTRO
APAGUEDESENHO	LIMPETELA
PARAFRENTE	FRENTE
PARATRÁS	VOLTE
PARADIREITA	DIREITA
PARAESQUERDA	ESQUERDA
USELÁPIS	COLORIDO
USENADA	SEMCOR
USEBORRACHA	—
MUDECL	COR
REPITA	REPITA
APRENDA	APRENDA
FIM	FIM
ADEUS	ADEUS

```
REPEAT 180 [FORWARD 1 RIGHT 1]
```

```
REPITA 180 [PARAFRENTE 1
PARADIREITA 1]
```

UMA PISTA DE CORRIDA

Agora que temos todos os "ingredientes" para montar um programa mais complexo, vamos tentar desenhar uma pista de corrida na tela. Para isso, dividiremos o problema em várias partes, ou subtarefas, que depois serão chamadas na ordem correta.

Começando com a parte interna da pista, definiremos um procedimento que trace a curva e, em seguida, a reta:

```
TO LADOINTERNO
FORWARD 100
REPEAT 180 [FORWARD 1 RIGHT 1]
END
```

```
AP LADOINTERNO
PF 100
REPITA 180 [PF 1 PD 1]
FIM
```

A combinação de duas curvas e duas retas nos fornecerá a parte interna da pista de corrida:

```
TO PISTAINTERNA
LADOINTERNO LADOINTERNO
END
```

```
AP PISTAINTERNA
LADOINTERNO LADOINTERNO
FIM
```

A parte externa da pista requer uma curva maior. Para obtê-la, aumentamos o passo da tartaruga:

```
TO LADOEXTERNO
FORWARD 100
REPEAT 90 [FORWARD 3 RIGHT 2]
END
```

e:

```
TO PISTAEXTERNA
LADOEXTERNO LADOEXTERNO
END
```

```
AP LADOEXTERNO
PF 100
REPITA 90 [PF 3 PD 2]
FIM
```

e:

```
AP PISTAEXTERNA
LADOEXTERNO LADOEXTERNO
FIM
```

Para começar a pista em um ponto adequado da tela, definimos o procedimento chamado INICIO:

```
TO INICIO
PENUP
LEFT 40 FORWARD 110 RIGHT 130
PENDOWN
END
```

```
AP INICIO
USENADA
PE 40 PF 110 PD 130
USELAPIS
FIM
```

Um outro procedimento, PARTIDA, moverá a tartaruga para onde a linha de partida e o início da parte interna da pista devem ser desenhados:

```
TO PARTIDA
RIGHT 90 FORWARD 30 LEFT 90
END
```

```
AP PARTIDA
PD 90 PF 30 PE 90
FIM
```

Finalmente, para traçar a pista toda de uma vez, definiremos um procedimento chamado PISTA.

```
TO PISTA
INICIO
PISTAEXTERNA
PARTIDA
PISTAINTERNA
END
```

Como posso traduzir os comandos de meu LOGO que estão em inglês?

Nada mais fácil. Como o LOGO é uma linguagem extensível, podemos desenvolver vários procedimentos simples, que obedecem a um comando em português e chamam o primitivo equivalente em inglês. Se você resolver, por exemplo, trabalhar com um BRASLOGO dentro do LOGO original, defina:

```
TO PARAFRENTE :D
FORWARD D:
END

TO PARATRAS :D
BACK :D
END

TO PARADIREITA :A
RIGHT :A
END

TO PARAESQUERDA :A
LEFT :A
END

TO PARACENTRO
HOME
END

TO REPITA :N :LIST
REPEAT :N :LIST
END
```

e assim por diante. Em seguida, grave esse conjunto de procedimentos com SAVE "BRASLOGO, e, quando quiser programar em português, carregue-o com LOAD "BRASLOGO.

```
AP PISTA
INICIO
PISTAEXTERNA
PARTIDA
PISTAINTERNA
FIM
```

Simples, não é?

Todos os programas em linguagem LOGO são elaborados dessa maneira. A divisão do programa principal em uma série de "tijolos" de construção torna bem mais fácil tanto a programação como a própria execução de testes. No próximo artigo, veremos como montar programas de maior complexidade usando essa mesma técnica.

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craft II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Craft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:


Sinclair ZX-81


TRS-80


TK-2000


MSX


Spectrum


TRS-Color


Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PERIFÉRICOS
Sistemas de controle por computador. Sensores e atuadores. Conexão ao micro. Software disponível.

LINGUAGENS
O LOGO e a tartaruga: procedimentos com variáveis. Desenho de círculos e polígonos. Repetição e recursão. Interrupção.

PROGRAMAÇÃO DE JOGOS
Compressão do texto de uma aventura. Substitutos para o código ASCII. O uso da estatística.

PROGRAMAÇÃO BASIC
Acesso às variáveis do sistema no Spectrum. Endereços úteis.